VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REZERVACE AUTOBUSOVÝCH JÍZDENEK Z MOBILNÍHO TELEFONU

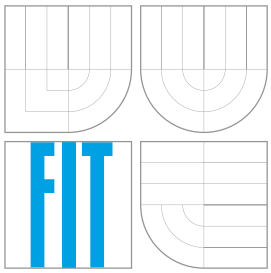
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ BENEŠ

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REZERVACE AUTOBUSOVÝCH JÍZDENEK Z MOBILNÍHO TELEFONU

RESERVATION OF BUS TICKETS FROM MOBILE PHONE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ONDŘEJ BENEŠ
AUTHOR

VEDOUCÍ PRÁCE — Ing. ADAM HEROUT, Ph.D.
SUPERVISOR

BRNO 2010

Abstrakt

Tato bakalářská práce se zabývá programováním pro mobilní platformu Android. Popisuje základní techniky programování pro tuto platformu. V rámci této práce byla vytvořena aplikace pro jednodušší rezervování autobusových jízdenek. Práce rozebírá konkrétní implementaci této aplikace.

Abstract

This bachelor's thesis deals with programming for mobile platform Android. It describes basic techniques of programming for this platform. As a part of this thesis was created an application for simplier bus ticket reservation. This thesis also analyses an implementation of this application.

Klíčová slova

Android, mobilní platforma, Java, SQLite, SMS

Keywords

Android, mobile platform, Java, SQLite, SMS

Citace

Ondřej Beneš: Rezervace autobusových jízdenek z mobilního telefonu, bakalářská práce, Brno, FIT VUT v Brně, 2010

Rezervace autobusových jízdenek z mobilního telefonu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Adama Herouta, Ph.D.

.....
Ondřej Beneš
18. května 2010

Poděkování

Především bych rád poděkoval mému vedoucímu Ing. Adamu Heroutovi, Ph.D. za trpělivost a cenné rady při konzultacích. Dále bych pak velký dík věnoval Ivě Tomalové za zpětnou vazbu a pomoc při korektuře této práce. V poslední řadě bych také poděkoval všem testerům aplikace.

Obsah

Kapitola 1

Úvod

Mobilní komunikace je dnes běžnou součástí našich životů. Od dob prvních mobilních telefonů už utekla spousta času a dnes je využíváme nejen jako komunikační nástroj. Běžné telefony díky barevným displejům používáme k pořizování fotografií, díky technicky vyspělým mobilním sítím můžeme využít připojení k Internetu k posílání e-mailů či poslední dobou stále víc populárním surfování po webových stránkách. Objevují se nové typy mobilních telefonů, které lze ovládat pouze dotykem, mobilní zařízení se pomalu integrují do automobilů či různých spotřebičů. Společně s technologickým pokrokem musí jít i vývoj softwaru na těchto zařízeních. Na trh přicházejí nové systémy pro tyto zařízení od velkých společností, komunity či od malých neznámých firem. Díky technologicky vyspělým zařízením jsou lidé schopni v reálném čase zjišťovat vlastní polohu a sdílet ji s ostatními, jsou schopni komunikovat například s bankou odkudkoliv, sdílet soubory či sledovat jízdní řády dopravních prostředků.

Jedním z nováčků na tomto trhu je platforma Android. Tato platforma letos oslaví teprve druhý rok, co spatřila světlo světa. Její možnosti jsou však víc než bohaté a stále se rozšiřují. Na Android se převážně vyvíjí jazyce Java, který si už našel své místo ve světě informatiky. Společně s kvalitní dokumentací tvoří jednoduchý nástroj pro tvorbu zajímavých aplikaci. Mě tato kombinace zaujala a rozhodl jsem se do světa Javy a Androidu nahlédnout. V době, kdy jsem volil téma mé závěrečné práce, mezi progamy pro platformu Android chyběla aplikace zjednodušující cestování autobusy společnosti Student Agency, a proto jsem se rozhodl vytvořit aplikaci pro snadnější rezervaci jízdenek.

V první kapitole je popsána samotná platforma Android, něco málo z její historie, jaké možnosti skýtá či kde všude se s ní můžeme setkat. Poslední sekce této kapitoly obsahuje stručný popis technik programování pro tuto platformu. Především se věnuji technikám, které byly použity v této práci. V této kapitole nezmiňuji programování v jazyce Java, přestože i ten pro mě byl stejně neznámým jako Android. Tento jazyk je všeobecně známý a rozšířený a jeho popis by znamenal spíše 'nošení dříví do lesa'.

Druhá kapitola rozebírá aktuální stav situace ohledně cestování a hlavně rezervování jízd u společnosti Student Agency. Popisuje také alternativní možnosti.

Třetí kapitola popisuje mnou navržené řešení a aplikaci. Jsou zde seskupeny jednotlivé komponenty aplikace podle jejich významu při vývoji.

Předposlední kapitola shrnuje informace o použití samotné aplikace a informuje o možné distribuci do světa.

V závěru pak hodnotím dosažené cíle práce.

Kapitola 2

Operační systém Android

Android je platforma pro mobilní zařízení vyvinuta společností Google.

2.1 Historie

Historie Androidu se začala psát roku 2005, kdy Andy Rubin a jeho společníci započali vyvíjet mobilní operační systém. O jeho vývoji se v té době moc nevědělo, na veřejnost proniklo pouze několik informací, že Google vstupuje do světa mobilních technologií.

V roce 2007 vzniklo uskupení Open Handset Alliance, které si kladlo za cíl vytvářet otevřená řešení pro mobilní zařízení. Mezi zakládajícími členy této aliance byly velké společnosti, které měly co dočinění s mobilními zařízeními. Mezi ně patří kromě Google například HTC, Intel, Texas Instruments, T-Mobile, Nvidia a spousty dalších. Prvním produktem této aliance byl právě Android, mobilní platforma založená na Linuxovém jádře verze 2.6 [9].

První verze Androidu se veřejně objevila na podzim roku 2008. Google zveřejnil zdrojové kódy systému pod Apache licencí. Od té doby se systém vyvíjí velkou rychlostí a Open Handset Alliance se stále rozrůstá o velké společnosti jako například Vodafone či Sony Ericsson.

2.1.1 Verze

Od svého vzniku systém prošel několika vývojovými verzemi. Každá z nich přidává nové možnosti systému a opravuje chyby, které se v průběhu používaní objevily. Jednotlivé verze jsou pojmenovávány podle nejrůznějších dezertů (viz 2.1).

1.5 Cupcake

Od vydání prvního Androidu uplynulo zhruba půl roku a v roce 2009 byla vydána nová verze systému 1.5 pod kódovým označením Cupcake. Nová verze přinesla spousty vylepšení, mezi hlavní patří softwarová klávesnice, změna vzhledu uživatelského rozhraní, možnost vkládání takzvaných widgetů na základní plochu či kopírování a vkládání textů v celém systému.

Chvíli poté byl vydán první vývojový kit (NDK[1]) umožňující vývoj nativních aplikací pro Android v jazycích C a C++.

[1] Native development kit

1.6 Donut

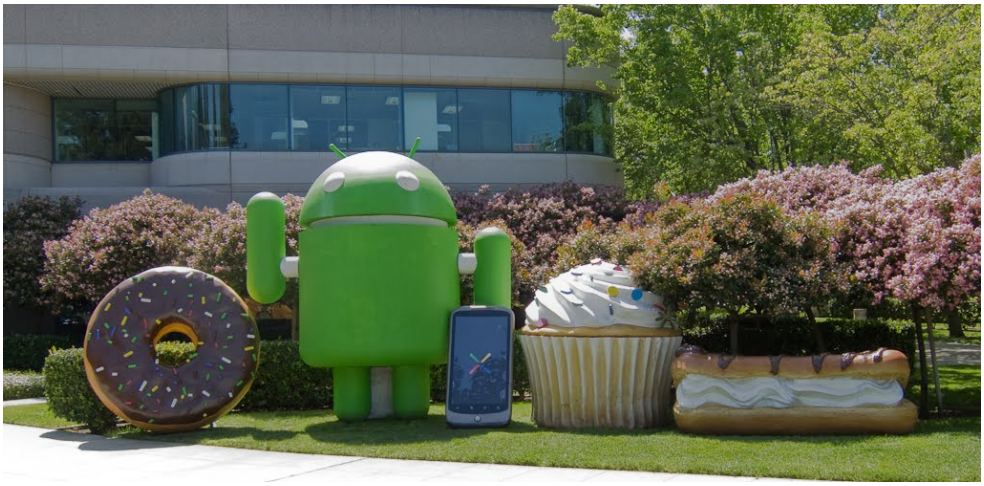
Koncem roku 2009, opět zhruba po půl roce od poslední verze Androidu, byla veřejnosti představena nová verze přinášející podporu nových technologií jako CDMA či VPN a spousty softwarových vylepšení.

Zároveň byla vydána další verze takzvaného kitu pro vývoj nativních aplikaci NDK a to revize číslo 2.

2.0/2.1 Eclair

Ještě tentýž rok, o měsíc později, byla vydána nová verze, a to první verze s hlavním číslem 2. V systému přibyla například podpora Bluetooth 2.1. Doznal opět spousty softwarových vylepšení, zrychlení uživatelského rozhraní, podpory HTML5 pro vestavěný prohlížeč či podpory Microsoft Exchange. Poprvé se zde také objevuje podpora vícedotykového ovládání.

Šest měsíců poté vyšla ještě nová a dosud poslední verze NDK.



Obrázek 2.1: Sochy jednotlivých verzí Androidu před Googleplexem v USA.

2.1.2 Android Market

Google k systému Android vytvořil i online trh s aplikacemi, kde si uživatelé mohou stahovat aplikace, které jsou zpoplatněné i zdarma. Vývojáři zde mají možnost své aplikace publikovat či dokonce prodávat. Toto však není v současné době umožněno ani zdaleka všem státům, kam Android pronikl. V těchto zemích (včetně České republiky) jsou k dispozici pouze aplikace zdarma a díky tomu jejich celkový počet převládá nad počtem placených. V současné době je na trhu k dispozici přes 40 000 různých aplikací. Přes trh se aplikace také aktualizují, pokud se objeví nová verze.

2.2 Možnosti systému

Možnosti využití platformy Android jsou velké a s každou novou verzí přicházejí další. V současné době jsou podporovány bezdrátové sítě jako například GSM, UMTS, Bluetooth nebo Wi-Fi. Platforma má podporu satelitní navigace GPS, podporu hardwarových čidel jako jsou akcelerometry, magnetometry či multimediální podporu různých audio či video

formátů. Podporuje i grafické knihovny pro 2D či 3D aplikace (OpenGL ES 1.0) či SQLite databáze.

2.3 Zařízení

Platforma Android je určena pro mobilní zařízení a proto její využití nejčastěji nalezneme v mobilních telefonech. Takováto zařízení produkují firmy HTC, Motorola či Samsung. Není to ale jediné možné využití této platformy. Další uplatnění našla u mininoteboků dnes známých jako netbooky či u elektronických čteček knih. Platforma se ale dostává i do neobvyklých zařízení jako jsou pračky, mikrovlnné trouby či palubní počítače automobilů.

2.4 Architektura systému Android

Architekturu systému Android lze rozdělit do 5 skupin (viz 2.2) [5].

Nejníže z nich je Linuxové jádro (*Linux Kernel*). To zajišťuje základní správu hardwaru jako je například správa paměti, správa procesů, systémové ovladače či síťová komunikace. Funguje jako abstraktní vrstva mezi hardwarem a zbylými softwarovými vrstvami.

Vrstva *Android Runtime* obsahuje základní knihovny poskytující funkčnost pro základní knihovny programovacího jazyka Java, pod kterým běží aplikace na Androidu.

Další vrstva *Libraries* obsahuje knihovny v jazyce C/C++ pro různé komponenty systému. Mezi základní knihovny patří například:

- **Systémové knihovny C** založené na BSD knihovně `libc`.
- **Multimediální knihovny** vkládající podporu pro mnoho různých multimediálních formátů jako například: MP3, MPEG4 či PNG.
- **Knihovny pro 2D a 3D zobrazení** včetně podpory OpenGL ES 1.0.
- **Knihovny pro podporu SQLite** jednoduché a rychlé relační databáze.
- **LibWebCore** knihovny pro podporu zobrazení moderních webových aplikací.

Nad vrstvou s knihovnami leží vrstva aplikačního frameworku (*Application Framework*). Tato část je k dispozici programátorům a obsahuje jednoduchý přístup k různým částem systému, jako například systém upozornění, správce telefonu či správce oken.

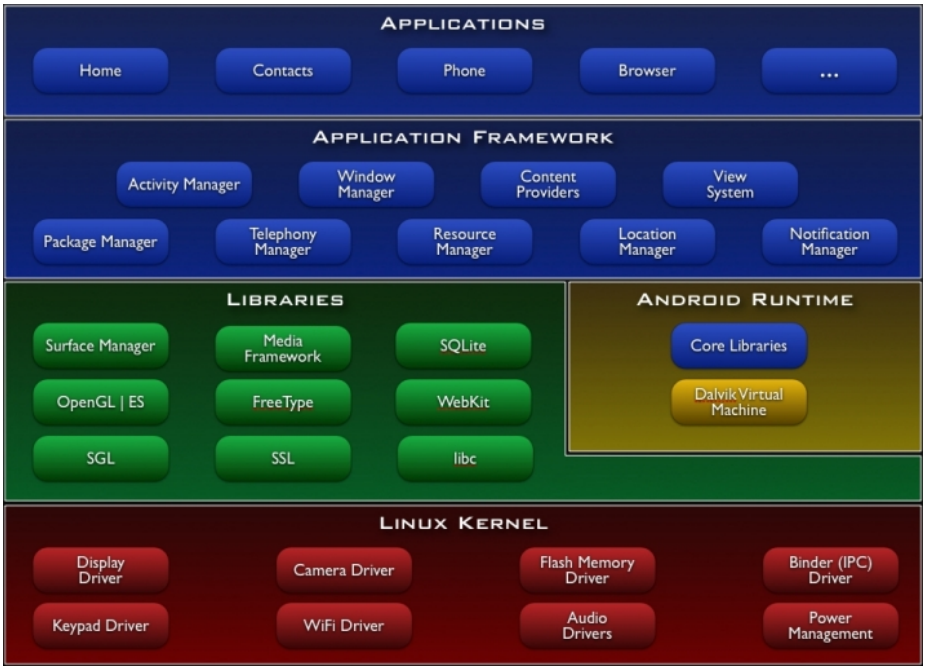
Poslední, nejvyšší vrstvou jsou samotné aplikace, které uživateli zpřístupňují systém v uživatelsky přívětivé podobě.

2.5 Programování pod Androidem

Pro vývojáře jsou v platformě připraveny dva různé způsoby, jak vyvíjet aplikace. Jeden umožňuje psát nativní aplikace přímo v programovacím jazyce C za pomocí nativního kitu. Tato možnost je určena pro vývojáře aplikací, u kterých závisí především na efektivnosti a rychlosti, jako například u her.

Dále je připraven SDK[2] kit umožňující vývoj aplikací v jazyce Java. Je zde připravena knihovna obdobná jazyku Java ME[3] obsahující některé základní knihovny jazyka Java

[2] Software Developement Kit
[3] Java Micro Edition



Obrázek 2.2: Architektura platformy Android

(upravené pro potřeby platformy Android) a další knihovny přímo pro platformu Android. Při spuštění aplikace se spustí samostatný proces. Každý proces má pak vlastní virtuální stroj Dalvik. Ten je obdobou virtuálního stroje JVM[4], avšak speciálně upravený pro potřeby v mobilních zařízení a také pro současný běh více těchto virtuálních strojů. V základu je každé aplikaci přiřazeno unikání idetifikační kód v systému a to z bezpečnostních důvodů. Data této aplikace jsou pak viditelná pouze pro ni samotnou. Součástí SDK je také emulátor kompletně simulující systém Android na počítači, který významně vývojářům zjednodušuje vývoj. Pro vývojáře používající SDK je zde také připraveno rozšíření pro vývojové prostředí Eclipse [3].

Tato práce se bude zabývat hlavně programováním za pomoci SDK.

2.5.1 Základní myšlenky vývoje

Jeden ze základních principů programování aplikací pro systém Android je možnost využití již napsaných aplikací či jejich fragmentů. Například pokud vytvoříme aplikaci pro fotografování a výslednou fotografii budeme chtít poslat e-mailem, nemusíme psát aplikaci pro odesílání e-mailů, ale jen fotku předáme systémové či jiné, pro to určené aplikaci. Aplikace tedy nemají jedno hlavní místo, kde začínají, ale jsou celé děleny do komponent, které se spouštějí dle potřeby. Mezi tyto základní komponenty patří aktivity, služby a přijímače broadcastů [1].

2.5.2 Aktivity

Aktivita je základní komponentou aplikací na Androidu. Představují uživatelské rozhraní pro aplikace. Jedna aktivita je v základu jedna obrazovka uživatelského rozhraní. Příkladem

[4] Java Virtual Machine

může být aplikace na psaní zpráv. Jedna obrazovka, která zobrazuje všechny zprávy, představuje jednu aktivitu, zatímco druhá obrazovka zobrazující samotnou zprávu představuje druhou aktivitu. Aplikace muže mít tedy jednu nebo mít více obrazovek. Je také možné mít aktivitu bez definovaného uživatelského rozhraní. Všechny aktivity jsou podtřídou třídy *Activity*.

Životní cyklus aktivity

Systém s aktivitami pracuje pomocí zásobníku aktivit. Ve chvíli, kdy se spustí jedna aktivita, vloží se na vršek zásobníku a stane se tak aktivní. Předchozí aktivity zůstávají pod ní v zásobníku a nestanou se aktivními, dokud aktivita na vrchu neskončí. Práci s tímto zásobníkem často ulehčuje hardwarové tlačítko Zpět, které je přítomno na většině zařízení, na kterých běží Android. Toto tlačítko odebírá z vrchu zásobníku aktuální aktivitu a tím navrací předchozí aktivitu do aktivního stavu.

Aktivity mají v systému určitý životní cyklus (viz 2.3) a mohou se dostat do několika stavů:

- Aktivita běžící (*running*) z vrcholku zásobníku na popředí je aktuálně viditelná.
- Pozastavená aktivita (*onPause*) je taková aktivita, přes kterou je transparetně zobrazena jiná aktivita, která se dostala na vrchol zásobníku. Zachovává si však stav a všechny informace jako běžící aktivita. Může být ukončena, pokud dojde k nedostatku paměti.
- Zastavená aktivita (*onStop*) je aktivitou, která je uživateli úplně skryta. Opět si ponechává stav a všechny informace jako aktivita běžící a jako pozastavená aktivita i tato aktivita může být v případě nedostatku paměti systémem ukončena.
- Ukončená aktivita (*onDestroy*) je taková, která vzniká, pokud v systému nejsou volné prostředky pro její běh.

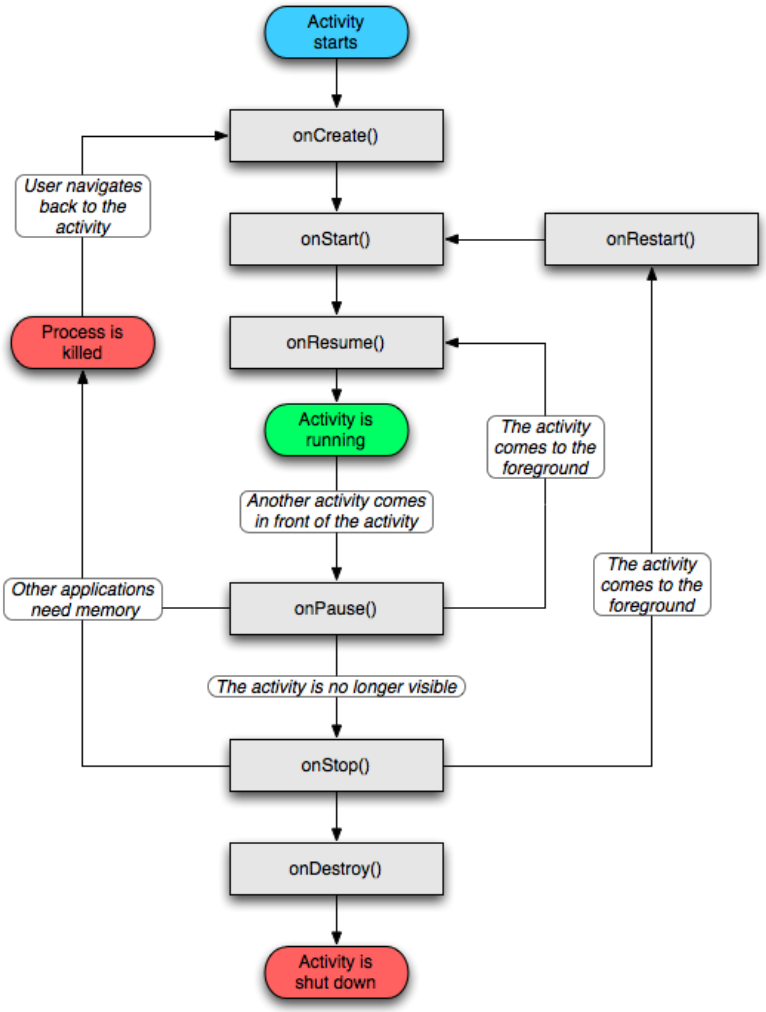
2.5.3 Sdělení

Jednotlivé komponenty systému musí mezi sebou nějakým způsobem komunikovat. V systému Android jsou k tomuto účelu využívány zprávy nazývané sdělení (v originále nazývány *Intents*). Sdělení slouží jak ke komunikaci mezi jednotlivými komponentami jedné aplikace, tak je lze využít ke komunikaci mezi komponentami jiných aplikací v rámci systému. Sdělení jako takové je pasivní objekt obsahující popis operace, která se má vykonat, nebo informace o tom, že došlo ke změně stavu systému. Používá se k aktivaci tří základních komponent systému a to aktivit, služeb nebo přijímačů broadcastů (viz 2.5.5).

Druhy sdělení

Sdělění se dělí do dvou základních kategorií:

- Explicitní - obsahují definici cíle, pro který je sdělení určeno, nejčastěji se používá pro komunikaci uvnitř aplikace
- Implicitní - neobsahují definici cíle a jsou používány pro aktivaci komponent jiných aplikací



Obrázek 2.3: Životní cyklus aktivity

Implicitní sdělení a filtry sdělení

Na rozdíl od explicitních sdělení, která mají přesně definovaný cíl, implicitní nejsou určena pro jednu konkrétní komponentu. Pro příjem implicitních sdělení je třeba u komponenty nadefinovat filtry sdělení. To, zda komponenta obdrží či neobdrží výsledné sdělení, určí systém na základě těchto filtrů. Pokud se nadefinované filtry shodují s poslaným sdělením, je dané komponentě sdělení předáno. Filtry se definují v `AndroidManifest.xml` souboru (viz 2.5.8), z kterého si je systém sám zjišťuje.

2.5.4 Uživatelské rozhraní

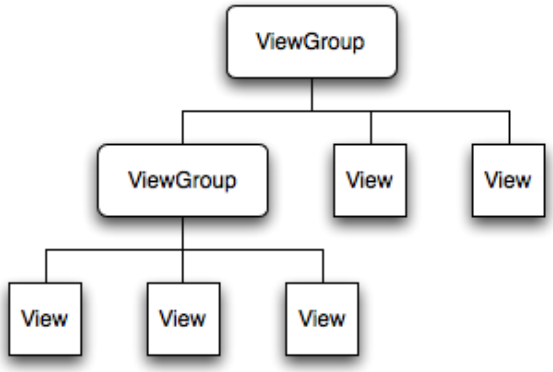
Uživatelské rozhraní se skládá z objektů odvozených od třídy *View* umístěných do hierarchického stromu. *View* objekt poskytuje základ pro podtřídy prvků uživatelského rozhraní (v originále nazývané *widget*). V systému je několik takových prvků předpřipraveno. Patří mezi ně například textová pole, tlačítka či seznamy. Tyto objekty se starají i o interakci s uživatelem, lze na ně klikat, měnit jejich obsah a podobně. Dalšími důležitým prvkem

uživatelského rozhraní a zároveň potomek třídy *View* je třída *ViewGroup* a její potomci. Tyto prvky se nazývají layouty a jsou to kontejnery pro jednotlivé *View* objekty [5].

Uživatelské rozhraní v Androidu lze vytvářet dvěma způsoby. Prvním z nich je vytváření běžně v kódu Java jako například AWT [8]. Jednotlivé prvky odpovídají objektům odvozených ze třídy *View*. Tato možnost zajišťuje dynamičnost uživatelského rozhraní a programátor má možnost za běhu programu vlastnosti uživatelského rozhraní měnit dle potřeb.

Druhou možností vytváření uživatelského rozhraní je definicí napsané v XML. Tento XML soubor se pak pomocí plniče layoutu (objekt *LayoutInflater*) vloží do kódu a vytvoří objektovou reprezentaci uživatelského rozhraní totožnou s tou, která se vytváří přímo v Java kódu. Jednotlivé vlastnosti prvků se nastaví pomocí atribut elementů XML. V dokumentaci k Androidu [1] lze u jednotlivých objektů dohledat jaké vlastnosti odpovídají jakým metodám. Pomocí XML definic lze vytvářet nejen layouty, ale i ostatní prvky uživatelského rozhraní, jako jsou například programové menu, či obrazovky s nastavením aplikace. Do aplikace se vloží pomocí podobných plničů jako u layoutu.

Kromě předpřipravených prvků uživatelského rozhraní lze vytvářet kompletně vlastní prvky. Prvky je možné vytvářet kompletně nové, včetně všech jejich vlastností, jako například letecká páka pro ovládání her či otočné kolečko pro ovládání hlasitosti. Je také možné vytvářet nové prvky skládáním jiných prvků. Ve výsledku vzniká objekt *View* a lze jej použít kdekoliv ve stromu uživatelského rozhraní (viz 2.4).



Obrázek 2.4: Strom objektů uživatelského rozhraní

2.5.5 Přijímače broadcastů

Přijímače broadcastů jsou komponenty, které nedělají nic jiného než, že reagují na broadcasty. Nejen systém, ale i různé aplikace mohou vysílat spousty broadcast informací (například zapnutí Wi-Fi, slabá baterie, nový e-mail a podobně), na které další aplikace reagují různě. Samotný přijímač nemá uživatelské rozhraní může však vyvolat aktivitu nějaké aplikace. Nejčastější akce, kterou přijímače dělají je zobrazení upozornění uživateli. Všechny přijímače rozšiřují abstraktní třídu *BroadcastReceivers* a musí implementovat jedinou abstraktní metodu *onReceive()*. Pro přijetí broadcastu je nutné se zaregistrovat v systému a to buďto pomocí metody *registerReceiver()* nebo staticky v souboru `AndroidManifest.xml` (viz 2.5.8) pomocí tagu `<receiver>`. Broadcasty se dělí na dvě skupiny:

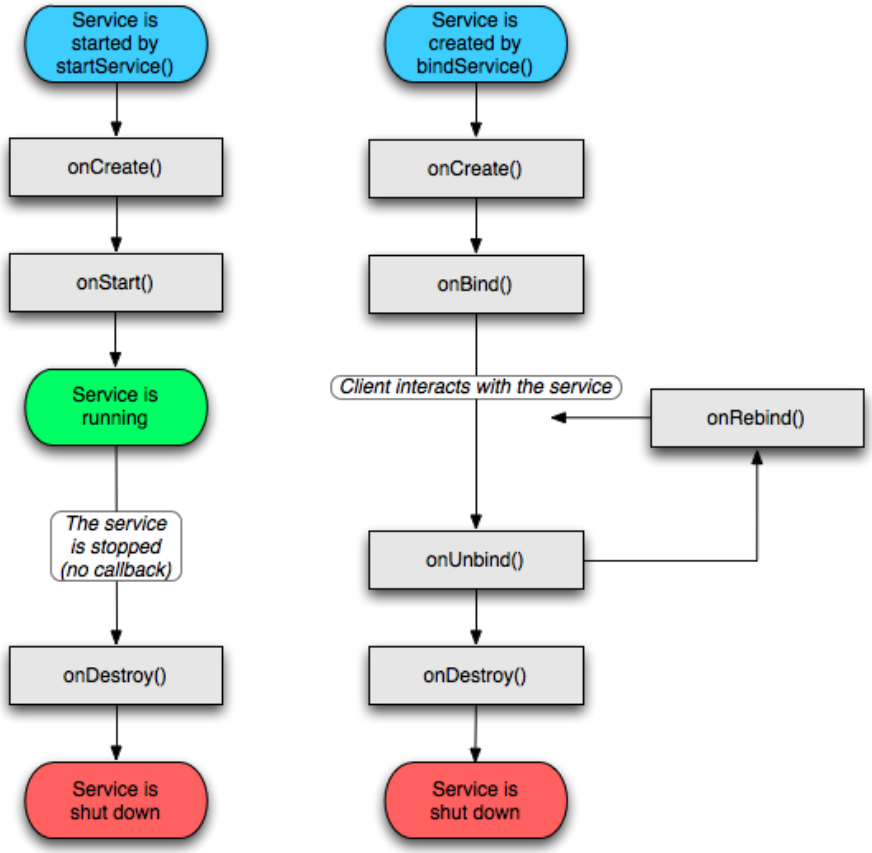
- **Normální broadcasty** jsou asynchronní a vysílají se nejčastěji všem přijímačům současně.

- **Seřazené broadcasty** jsou přijímány postupně jedním přijímačem. Ten je může předat dalším, nebo může další vysílaní zrušit. Přijímačům lze nastavit priority.

Životní cyklus přijímače je jednoduchý. Přijímač je aktivní pouze, pokud je v metodě *onReceive()*, po jejím ukončení se stává neaktivním.

2.5.6 Služby

Pro dlouhodobý běh operací má Android služby (v originále *Services*). Tyto služby běží na pozadí bez uživatelského rozhraní. Jsou využívány hlavně pro běh operací, které běží kontinuálně. Příkladem takové služby je přehrávání hudby. Přehrávač při spuštění hudby spustí službu, která zajišťuje přehrávání z reproduktoru. Tento přehrávač se pak připojí na tuto službu a ovládá jednotlivé stavy přehrávání, jako například další skladba či ovládání hlasitosti. Pokud se přehrávač vypne, hudba bude hrát dál, dokud služba zajišťující přehrávání neskončí nebo nebude systémem ukončena.



Obrázek 2.5: Životní cyklus služby

Jak již bylo zmíněno, ke službám lze přistupovat dvěma způsoby:

- Služby lze spustit pomocí *startService()* metody a následně pak ukončit pomocí jedné z ukončovacích metod.
- Na již běžící službu se lze připojit pomocí *bindService()* a následně odpojit pomocí *unbindService()*.

Služba má podobně jako aktivity (viz 2.5.2) životní cyklus. Není tak komplexní jako u aktivity, za to má dvě možnosti. První odpovídá samotnému spuštění služby a druhá odpovídá stavu po připojení na již běžící službu. Cykly lze vidět na obrázku 2.5.

2.5.7 Poskytovatelé obsahu

Data mezi jednotlivými aplikacemi je možné sdílet. V základu nejsou data jedné aplikace k dispozici žádné jiné z bezpečnostních důvodů, ale poskytovatelé obsahu (v originále *Content Providers*) umožňují vývojářům sdílet data z vlastních aplikací ostatním aplikacím. Příkladem je možnost čtení kontaktů ze základní aplikace. Lze získat kontakty včetně telefonů či adres a není nutné do aplikace implementovat vlastní seznam. Přístup k datům je zprostředkován pomocí objektu *ContentResolver*. K datům se přistupuje pomocí speciálních URI[5], které jsou poskytovatelem obsahu předem nastaveny. Pomocí těchto URI se vytvoří dotaz na data a systém vrátí objekt *Cursor*. Tento objekt poskytuje rozhraní pro přístup ke čtení či zápisu k množině dat poskytované dotazem z databáze [5]. Pomocí objektu *Cursor* se dá v datech procházet tam i zpět po jednotlivých řádcích. Ukázka, jak mohou vypadat vrácená data, jsou v následující tabulce:

_ID	ČÍSLO	JMÉNO
1	+420737123456	Jan Novák
2	+420737333444	Petr Náprstek
3	+420606557744	Lucie Novotná
4	+420777987654	Josef Zelený

S daty se pracuje jako s ostatními daty přístupnými přes objekt *Cursor*. Je tedy možné data nejen číst, ale také vytvářet, upravovat či mazat.

Pokud chceme data vlastní aplikace nabídnout i ostatním aplikacím, musíme si vytvořit vlastního poskytovatele obsahu. Ten rozšiřuje abstraktní třídu *ContentProvider* a implementuje všechny abstraktní metody.

2.5.8 AndroidManifest.xml

Každá aplikace pro Android musí v kořenovém adresáři obsahovat `AndroidManifest.xml`, soubor, který v sobě obsahuje důležité informace pro její běh v systému [5]. Soubor je ve formatu XML [6]. Některé informace obsažené v souboru:

- Název Java balíčku, který je unikátní v systému.
- Popis jednotlivých komponent aplikace, jako jsou aktivity (viz 2.5.2), služby (viz 2.5.6), přijímače broadcastů (viz 2.5.5) či poskytovatelé obsahu (viz 2.5.7).
- Uvedení, jaké sdělení (viz 2.5.3), jsou kde použitelné pomocí filtrů sdělení.
- Deklarace povolení k přístupu k jednotlivým částem systémového API jako například odesílání SMS.
- Určení, jaké povolení je nutné mít k používání jednotlivých komponent aplikace ostatními aplikacemi.

[5] Uniform Resource Identifier
[6] Extensible Markup Language

- Definice minimální verzi Android API, kterou aplikace vyžaduje pro svůj běh.

Pokud v tomto souboru není komponenta, kterou chceme využít, nadefinována, není možné ji použít.

Kapitola 3

Analýza problému

Tato kapitola rozebírá problematiku rezervace jízdenek u společnosti Student Agency, stávající možnosti a různé způsoby jejich řešení.

3.1 Cestování se Student Agency

Společnost Student Agency provozuje autobusovou dálkovou dopravu na několika linkách na území České republiky a několika linkách do zahraničí. Oproti většině autobusových dopravců, je cestující povinnen rezervovat si místo v autobuse, kterým chtějí cestovat. Rezervace je spojená se zaplacením cesty a způsoby jak uhradit takovou rezervaci jsou následující:

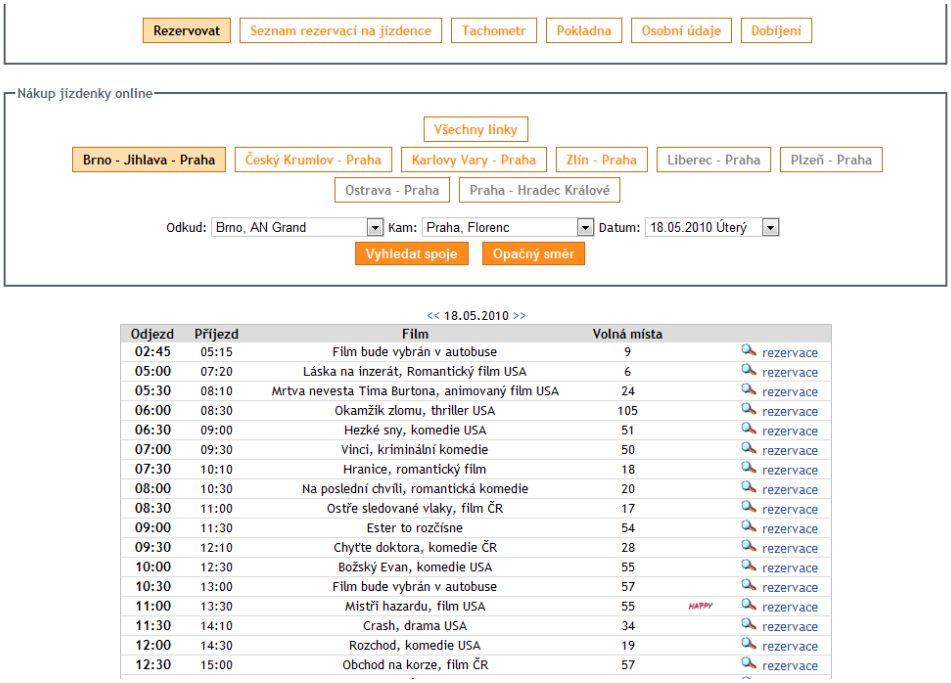
- Prvním způsobem je jednorázová jízdenka na pevné datum, která se zakoupí na konkrétní trasu a čas na prodejních místech.
- Nejčastěji využívaným způsobem platby je takzvaná otevřená jízdenka. Tu si cestující zakupuje taktéž předem na prodejních místech Student Agency a předplatí libovolnou částkou. Jízdenka je s unikátním číselným kódem. Tato jízdenka není vystavena na konkrétní trasu, datum ani čas. Je omezena pouze platností ceníku na dané trase.
- Třetí způsob je podobný předchozímu způsobu s tím rozdílem, že cestující si nepředplácí jednotlivé jízdenky, ale vlastní takzvanou kreditovou jízdenku, kterou si u společnosti zřídí. Je také opatřena unikántním číselným kódem pro rezervace a navíc při jejím zřízení jsou cestujícímu přiděleny unikátní přihlašovací údaje pro webový informační systém.
- Poslední způsob úhrady jízdného je takzvaná elektronická jízdenka. Od otevřené jízdenky se liší pouze v tom, že je možné ji zakoupit pouze na webových stránkách společnosti Student Agency.

Kromě jízdenky na pevné datum, všechny typy jízdenek spojuje způsbo rezervace vybraného spoje pomocí unikátního číselného kódu [10]. Samotnou rezervaci je možné provést dvěma způsoby.

3.1.1 Webový systém společnosti Student Agency

Prvním způsobem je webový rezervační systém společnosti Student Agency (viz 3.1). V případě kreditové jízdenky se cestující přihlásí a zarezervuje si v systému spoj. Při použití

otevřené nebo elektronické jízdenky se uživatel do systému přihlásí pomocí unikátního číselného kódu.



Obrázek 3.1: Webový rezervační systém společnosti Student Agency

3.1.2 Rezervace pomocí SMS

Druhá možnost rezervace spoje je pomocí SMS zpráv. Tato varianta je bezplatná ze strany Student Agency, jediné co je zpoplatněno je odeslaní SMS zprávy. Společnost disponuje telefonním číslem, na které lze posílat SMS zprávy v určitém, poměrně složitém formátu 3.1.2, který umožňuje základní správu jako je vytváření či mazání rezervací. Je zde také možnost zjišťování obsazenosti spojů. Tato možnost rezervace je funkční pouze na vnitrostátní spoje. Výjimku tvoří spoje na Slovensko a do Vídně, které lze také rezervovat pomocí SMS zpráv.

Příklady formátů SMS zpráv

Rezervace

```
BUS XXXXXXXXXXX DDMM HHMM AA BB S
```

Zrušení rezervace

```
BUS zrus XXXXXXXXXXX DDMM HHMM AA BB S
```

Zjištění volných míst ve spoji

```
BUS MISTA DDMM HHMM AA BB
```

Výpis rezervací a kreditu

```
BUS VYPIS XXXXXXXXXX
```

Vyhledávání odjezdů

```
BUS RAD DDMM AA BB
```

Legenda

XXXXXXXXX - číslo čárového kódu
DDMM HHMM - určení data a času rezervace
AA - určení výchozí stanice
BB - určení cílové stanice
S - číslo sedadla

Kódy jednotlivých stanic

Každá autobusová zastávka má své unikátní dvoumístné znakové označení. S expanzí společnosti Student Agency na českém trhu se rozrůstá jejich počet a stávájí se tak nezapamatovatelné pro obyčejného člověka, což znemožňuje rezervaci pomocí SMS nezávisle na dalších informačních zdrojích, jak byla původně pravděpodobně zamýšlena. Konkrétní kódy zastávek jsou k dohledání na webových stránkách společnosti (viz. [11]).

3.1.3 Jízdní řády

Jelikož se jedná o společnost dopravní, jízdní řady jsou jejich nepostradatelnou součástí. Pro získání informací o spojích této společnosti existuje několik způsobů:

- První a asi nejpoužívanější je možnost přečíst si jízdní řády jednotlivých linek na webových stránkách. Společnost poskytuje HTML[1] stránky s jízdním řádem.
- Druhá možnost je pomocí SMS zprávy 3.1.2 zjistit, jaké jsou v konkrétní den volné spoje, společnost zpětně zdarma pošle časy odjezdů dané linky.
- Třetí možnost je zjištění odjezdů osobně na zastávkách jednotlivých spojů či v kontaktních centrech společnosti.

Bohužel společnost neposkytuje žádné API[2]. Z těchto možností jsem zvolil jako hlavní zdroj dat pro jízdní řády možnost první. Kontaktoval jsem společnost Student Agency, zda mají zájem o spolupráci, ale bohužel zájem neměli a nepovolili ani automatické získávání dat z HTML kódu. Do aplikace jsem tedy musel ručně přepsat jízdní řády do formátu XML, který v aplikaci používám.

3.2 Obdobné aplikace

K tomuto projektu existují již obdobné aplikace pro platformu Android, které mají podobný princip a určení.

[1] Hypertext Markup Language - značkovací jazyk pro psaní webových stránek
[2] Application programming interface - rozhraní pro programování aplikací

3.2.1 SMS jízdenka

V několika městech v České republice je v současné době možnost získat místo běžné papírové jízdenky na městkou hromadnou dopravu jízdenku ve formě SMS zprávy. Rezervace funguje tak, že cestující odešle speciální řetězec v SMS zprávě na určené číslo a obratem dostane SMS zprávu s unikátním kódem, který slouží jako jízdenka. Pro platformu Android pro tento systém vytvořila společnost Inmite s.r.o. zjednodušující aplikaci. Aplikace umožňuje poslat požadavek na SMS jízdenku bez znalosti speciálních řetězců či cílových čísel. Odpověď ve formě SMS zprávy je aplikace schopná odchytit a přečíst. Cestujícímu je tak poté schopna zobrazit zbývající čas do konce jízdenky nebo jej upozornit před vypršením. Aplikace funguje v několika městech České republiky a nyní dokonce i v některých městech na Slovensku [14].

3.2.2 Student Agency SMS Rezervace pro Android

V době psaní této práce se na Android Marketu objevila aplikace umožňující jednodušší rezervaci jízdenek pomocí SMS zpráv u společnosti Student Agency. Tato aplikace má jednu obrazovku a prakticky jen uživateli umožňuje jednoduší zadávání SMS zprávy [13].

Kapitola 4

Návrh a implementace

Následující kapitola se věnuje implementaci samotné aplikace pro rezervování jízdenek pro Android. Je zde popis jednotlivých tříd, jak zděděných od systémových, tak tříd nově vytvořených pro potřeby aplikace. V následujících kapitolách jsou třídy použité v aplikaci popsány.

Aplikace prošla několika návrhy uživatelského rozhraní, pokaždé byly zjištěny různé nedostatky v použitelnosti a výsledné použité uživatelské rozhraní vyšlo jako nejlépe použitelné. Prvně jsem navrhl na papír, jak by uživatelské rozhraní mohlo vypadat a chovat se, podle toho jsem následně promyslel strukturu tříd a komponent samotné aplikace.

4.1 Vlastní třídy

Kromě základních tříd pro tvorbu aplikace jsem vytvořil několik podpůrných tříd, které zajišťují jednodušší interakci mezi jednotlivými třídami uživatelského rozhraní a jejich komponenty.

Třída Connection

Téměř celá aplikace využívá třídu Connection. Tato třída reprezentuje jedno spojení. Třída umožňuje získání základních informací o spojích, jako například času a místa. Poskytuje také základní formáty času a dat pro ostatní třídy. Data uložená v objektu jsou reprezentována pomocí rozhraní *Calendar* [6].

Třída City

Tato třída je jen pomocná třída pro jednodušší práci se základními daty o městě ze zdrojových souborů.

Třída SMS

Vlastní třída SMS je jen obalovou třídou pro práci se zprávami SMS. Obaluje základní systémové třídy *SmsManager* pro odesílání zpráv a zároveň rozšiřuje tuto možnost o odesílání konkrétních zpráv ve formátu, který požaduje společnost Student Agency u SMS rezervace (viz 3.1.2). Je zde definováno několik typů zpráv, které je možné odeslat:

- **TYPE_RESERVATION** odpovídá typu pro rezervaci konkrétního spoje

- **TYPE_CANCEL_RESERVATION** je typ, který zruší již existující rezervaci
- **TYPE_CHECK_CONNECTION_SEATS** zpráva pro kontrolu obsazenosti konkrétního spoje
- **TYPE_CREDIT_INFO** odešle dotaz na stav kreditu a rezervací na kreditové či jiné jízdence zadané kódem
- **TYPE_TIMETABLE** SMS dotazující se na stav jízdního řádu v konkrétní den

4.2 XML parsery

Jak je uvedeno výše, společnost Student Agency odmítla spolupráci, a protože nemají k dispozici žádný veřejný zdroj dat jízdních řádů, rozhodl jsem se tyto údaje získat extrakcí z webových stránek do souborů `timetable.xml` a `cities.xml`, které jsou přiloženy ke zdrojovým kódům. Při prvním spuštění aplikace se tyto soubory načtou a získají se z nich data do SQLite databáze, která je následně využívána pro zobrazování dat. Tuto operaci zajišťují třídy *XMLParseConnections* a *XMLParseCities* rozšiřující základní třídu *DefaultHandler*, které pomocí SAX [1] knihoven umístěných přímo v Androidu parsují data ze souborů. Při získávání a ukládání dat jsou využívány třídy *City* a *Connection*.

4.3 Aktivity

Základní třídy celé aplikace jsou aktivity (viz 2.5.2) rozšiřující třídu *Activity*. Ty zajišťují zobrazení uživatelského rozhraní. V aplikaci jich je použito celkem šest. Jako hlavní aktivita, ze které se spouští ostatní aktivity, je třída *ZlutyActivity*.

Aktivita ZlutyActivity

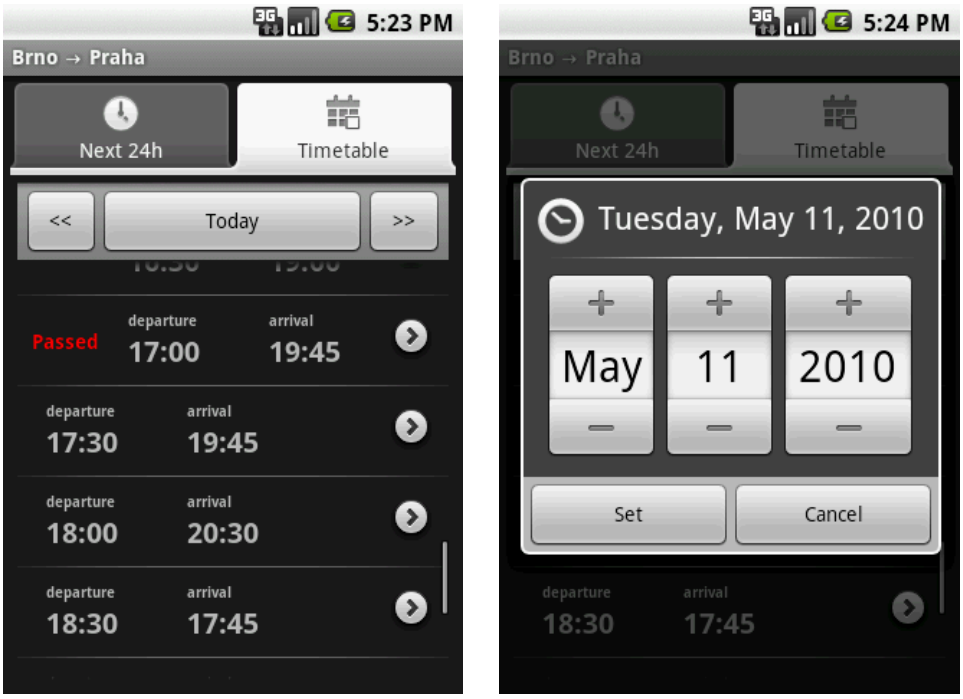
Jako u ostatních aktivit je v aktivitě *ZlutyActivity* metoda *onCreate()*, která se spustí jako první. Prvně se nastaví layout aktivity, který je definovaný v XML souboru, a to pomocí metody *setContentView()*. Poté se inicializuje databázový adaptér. Dále následuje inicializace dat pro jednotlivé prvky uživatelského rozhraní. Jsou zde uvedeny některé vnitřní třídy, nejčastěji to jsou naslouchače událostí, jako například *OnItemSelectedListener()* či *onItemClick()*. Ty obsluhují konkrétní událostí prvků uživatelského rozhraní. Dále jsou zde metody pro inicializaci hlavního menu této aktivity a obsluhy událostí vyvolaných tímto menu. Metoda *onDialogCreate()* zajišťuje správu dialogů, které vyvolává hlavní menu aktivity. Nakonec jsou zde pomocné metody, které získávají data z databáze na základě jednotlivých stavů uživatelského rozhraní. To obsahuje jeden *Spinner* pro výběr startovní stanice a *ListView* pro zobrazení cílových destinací z dané lokace. Kliknutím na položku v seznamu cílových míst je vyvolána nová aktivita a to aktivita *ConnectionsActivity*.

Aktivita ConnectionActivity

Aktivita *ConnectionsActivty* rozšiřuje třídu *TabActivity* a zajišťuje zobrazení obrazovky se záložkami. Obsah záložek je možné definovat přímo v této aktivitě a nebo je možné jej plnit jinými aktivitami. Zvolil jsem druhou možnost, a to plnění obsahu jednotlivých

[1] Simple API for XML

záložek jinými aktivitami. V této třídě je ve výsledku tedy jen definice jednotlivých záložek a přiřazení obsahu je zde provedeno pomocí sdělení (viz 2.5.3). Definiční soubor uživatelského rozhraní v XML ve výsledku obsahuje tedy jen informace o tom, že layout obsahuje záložky [2].



(a) Aktivita *TimetableActivity*

(b) Dialog pro výběr data

Obrázek 4.1: Ukázka obrazovky druhé záložky s aktivitou *TimetableActivity* a systémového dialogu pro výběr data

Aktivita NewestActivity

Obsah první záložky zajišťuje aktivita *NewestActivity*. Ta zobrazuje spoje v následujících 24 hodinách na dané lince. Je zde pouze jeden *ListView*, který zobrazuje konkrétní spoje. Metody v této aktivitě jsou obdobné jako u aktivity *ZlutyActivity*. Navíc jsou tu metody zajišťující vytvoření kontextového menu k jednotlivým položkám seznamu spojení a zpracování událostí jím vyvolaných. Nově zde bylo v metodě *onCreate()* použito vlákna [7]. Byla použita, protože získávání dat z databáze je pomalejší a blokuje hlavní vlákno, které zajišťuje vykreslování uživatelského rozhraní. Načítání dat bylo zajištěno pomocí třídy *Handler*, které je zodpovědná za posílání zpráv ve vláknech mezi sebou a hlavně pomocí samotného nového vlákna zajišťujícího načítání dat z databáze. V průběhu načítání dat je nad aktivitou zobrazen *ProgressDialog*. Data seznamu poskytuje adaptér *ConnectionsAdapter*.

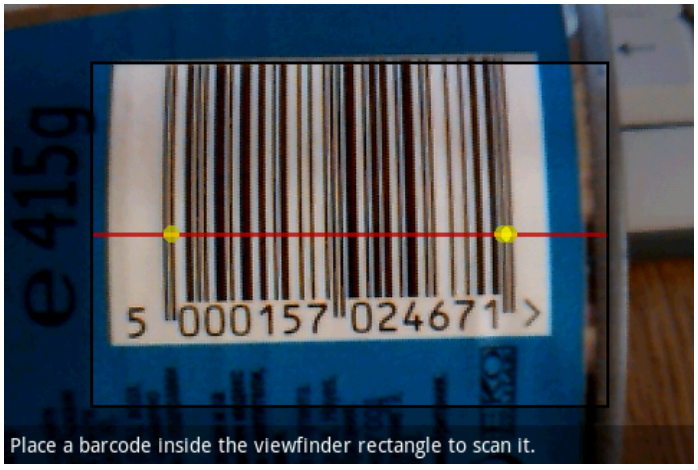
Aktivita TimetableActivity

Druhá záložka je vyplněna třídou *TimetableActivity*. Obsah je podobný aktivitě *NewestActivity*. Tato aktivita zobrazuje spoje po dnech (viz 4.1a). Mezi jednotlivými dny lze listovat pomocí tlačítek umístěných nad seznamem. Je zde také možnost vybrat konkrétní datum

pomocí systémového dialogu pro výběr data. Nově je zde metoda *onPrepareDialog()*. Ta zajišťuje při opětovném otevření dialogu (viz 4.1b) pro výběr data, to, že daný dialog obsahuje správné aktuální datum. Pokud je spoj již historií, je označen červeným nápisem. Při každé změně data, se data v seznamu obnoví. Plnění dat do seznamu zajišťuje stejně jako u aktivity *NewestActivity* adaptérová třída *ConnectionsAdapter*.

Aktivita BookActivity

Seznamy se spojením umístěné v jednotlivých záložkách *ConnectionsActivity* přes kontextové menu mohou vyvolat aktivitu *BookActivity*. Tato aktivita se stará o rezervování konkrétního spoje. Uživatel zde doplní informace vyžadované k samotné rezervaci, přičemž některé jsou volitelné. První povinná informace, kterou je třeba vyplnit, je kód pro rezervaci. Je zde více možností vložení kódu. Uživatel může opsat číslo přímo z lístku za pomoci softwarové či hardwarové klávesnice (závisí na zařízení) nebo je možné použití kódu kreditové jízdenky uloženého v nastavení aplikace. Pokud uživatel nezadal hodnotu, je upozorněn dialogem s možností vstupu do nastavení a zadání kódu. Jako poslední možnost jsem využil vlastnost Android systému, a to sdílení komponent aplikací. Konkrétně se jedná o možnost naskenování čárkového kódu jízdenky fotoaparátem pomocí aplikace Barcode Scanner. Do aplikace jsem vložil dvě předpřipravené třídy *IntentIntegrator* a *IntentResults*, které obalující základní práci se zíkáním výsledku skenování pomocí sdělení (viz 2.5.3). Ve vlastní třídě jsem pak zavolal pouze statickou metodu, která se o naskenování postará [12], a to i v případě, že aplikace na skenování není v zařízení přítomna. Je to užitečná a praktická ukázka spolupráce aplikací na Androidu.



Obrázek 4.2: Získání kódu naskenováním pomocí fotoaparátu

Aktivita PreferencesActivity

Z některých aktivit lze vyvolat menu, kde jedna z položek je vstup do nastavení aplikace. Tato položka vyvolá aktivitu *PreferencesActivity*. Ta je nadstavbou jedné z možností ukládání dat, a to *Preferences* [5]. Ty představují jednoduchou formu ukládání dat pro aplikaci. Jak název napovídá, jejich nejčastější použití je ve formě nastavení. Hodnoty jsou ukládané ve formě klíč-hodnota a jsou přístupné kdekoliv v aplikaci přes *Context* metodou *getSharedPreferences()*. Obrazovka nastavení je také načítána z XML souboru, kde je defi-

nována. Samotná aktivita dědí *PreferenceActivity* a v jediné metodě *onCreate()* je jen načtení *Preferences* z XML. Zobrazení, ukládání a ostatní práce jsou zajištěny systémem.

4.4 Adaptéry

V aplikaci se objevují prvky obsahující různá data. Správu a přístup k těmto datům zprostředkovávají adaptéry. Jsou také zodpovědné za zobrazení *View* objektu zobrazujícího data [1]. V aplikaci je použito hned několik adaptérů.

Databázový adaptér

O přístup k datům uloženým v SQLite databázi se stará třída *DbAdapter*. Tato třída obsahuje jednu vnitřní třídu *DatabaseHelper*, která rozšiřuje abstraktní třídu *SQLiteOpenHelper*, jež se stará o vytvoření databáze a její prvotní naplnění. Systém umožňuje databázi verzovat pro případné budoucí změny. Abstraktní třída obsahuje metodu *onUpgrade()*, která se aktivuje, pokud v kódu je použita nová verze databáze. Lze zde tedy možnost upravit starší strukturu databáze na novější bez ztráty dat v případě, že to návrh umožňuje. Vnitřní třída za pomocí tříd na parsování XML (viz 4.2) získá data z přiložených souborů a za pomoci vlastních tříd (viz 4.1) data vloží do databáze.

V adaptéru samotném jsou pak k dispozici různé metody pro získání dat z databáze. Většina z nich vrací data přes objekt *Cursor* pro jednotnou správu těchto dat. Metody pro získávání dat mají prefix `fetch`.

ArrivalPlacesAdapter - adaptér příjezdů

Na úvodní obrazovce se zobrazuje seznam cílových stanic, který je plněn adaptérem *ArrivalPlacesAdapter*. Tento adaptér rozšiřuje základní abstraktní třídu *BaseAdpater*. Ta slouží jako základ pro všechny adaptéry plnící *ListView* či *Spinner* objekty. *BaseAdapter* třída implementuje mimo jiné rozhraní *Adapter*, které vyžaduje implementaci základních metod pro práci s množinou dat, jako je získání aktuálního prvku, získání celkového počtu prvků či získání *View* objektu. Poslední zmíněná metoda je důležitá z pohledu efektivnosti výsledného seznamu, a proto bylo nutné se na ní podívat podrobněji. Účel této metody je získání objektu *View* konkrétního prvku množiny.

Metoda *getView()* obsahuje tři parametry [1]:

- **int position** Obsahuje pozici prvku v množině, pro který chceme *View* objekt.
- **View convertView** Obsahuje starý *View* objekt, který již není na obrazovce viditelný. Pokud není objekt nulový, je možné tento objekt znovu použít a tím urychlit vykreslování uživatelského rozhraní. Pomocí tohoto triku je možné dělat téměř nekonečné seznamy, které je možno procházet rychle a efektivně. Tento trik se dá nazvat recyklací objektu. Pokud však není k dispozici starý nepoužívaný *View* objekt, je nutné vytvořit nový.
- **ViewGroup parent** Rodičovský objekt, pro který bude případně tento objekt použit.

V těle této metody po inicializaci *View* objektu nastává naplnění daty z listu poskytovaného aktivitou.

Objekt ArrivalView

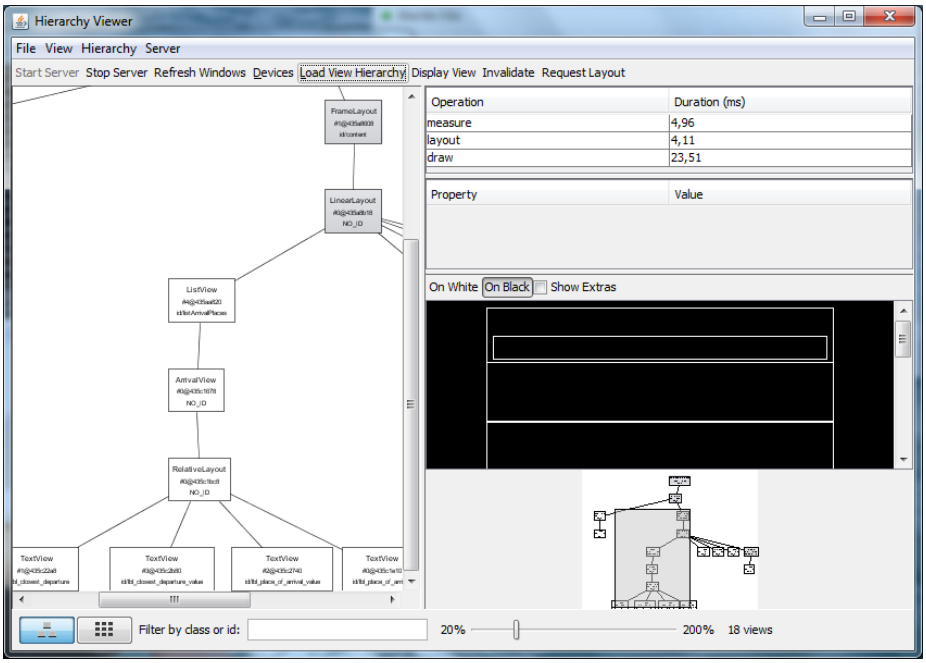
Adaptér v metodě, která získává *View* objekt, využívá objektu *ArrivalView*. Tento objekt reprezentuje prvek uživatelského rozhraní, který bude vložen na místo prvku seznamu. Podobně jako layoutů aktivit (viz 2.5.2) je i zde možnost tento objekt naplnit pomocí metody *inflate()* definicí z XML souboru a i zde to bylo takto použito. Je zde připravena metoda, která umožňuje změnit text uvnitř zkomponovaného nového *View*.

ConnectionsAdapter - adaptér pro spojení

Obdobně jako pro úvodní obrazovku s cílovými destinacemi, je pro další obrazovky s konkrétními odjezdy na dané lince vytvořen adaptér, který se stará o plnění dat, jak na obrazovce nejbližších spojení (viz 4.3), tak i na obrazovce s jízdním řádem (viz 4.3). I zde probíhá, stejně jako u adaptéru pro odjezdová místa, takzvaná recyklace *View* objektu pro efektivnější a rychlejší zobrazení. Navíc zde probíhá kontrola konkrétního spoje, zda již neodjel. Pokud tak nastane, u spoje se objeví červené varování.

Objekt ConnectionView

I u adaptéru ConnectionsAdpater je vlastní *View* objekt, který už je o něco složitější než u adaptéru *ArrivalPlacesAdapter*. Umožňuje nastavit popisek, který je v základu skryt, umožňuje také upravit časy odjezdů a zobrazuje jednoduché sdělení o spoji.



Obrázek 4.3: Utilitka HiearchyViewer pro inspekci uživatelského rozhraní

4.5 Uživatelské rozhraní

Jednotlivé části uživatelského rozhraní u Androidu lze definovat pomocí XML. To má hned několik výhod.

Hlavní z nich je oddělení definice uživatelského rozhraní od výkonného kódu. Aplikaci to významně zpřehlední.

Další z výhod je možnost náhledu na uživatelské rozhraní. Vývojáři Androidu vytvořili pro vývojové prostředí Eclipse rozšíření umožňující jednodušší vývoj pro platformu Android využívající utilitky[2] z SDK pro Android. Obsahuje mimo jiné i možnost náhledu uživatelského rozhraní definovaného v XML. Tento náhled je statický, nelze v něm s jednotlivými objekty *View* manipulovat, jak je tomu v jiných vývojových prostředích. Pro mobilní projekty je ale tento náhled víc než dostačující. Pro testování, jak uživatelské rozhraní interaguje, je k dispozici kompletní emulátor platformy Android pro počítač, který je součástí SDK.

Pro inspekci uživatelského rozhraní, jak jsou jednotlivé *View* objekty uspořádány na vykreslené obrazovce, jsem použil utilitku *HiearchyViewer* (viz 4.3), který je součástí SDK pro Android. Utilika umožňuje sledovat, jak rychle se uživatelské rozhraní vykresluje a hlavně, jaký strom objektů *View* a *ViewGroup* má výsledné uživatelské rozhraní.

XML layouty

V projektu jsou použity dva layouty, které obsahují ostatní *View* objekty.

Prvním z nich je *LinearLayout*. Tento potomek třídy *ViewGroup*, jak již sám název napovídá, umísťuje prvky rozhraní lineárně. Lze zvolit, zda to bude horizontálně či vertikálně. Jako prvek tohoto layoutu může být také jiný *LinearLayout*. Vhodnou kombinací tedy lze udělat uživatelské rozhraní dle libosti. Nevýhoda tohoto řešení je, že pokud chceme vytvořit trošku složitější kompozici prvků, výsledek obsahuje příliš mnoho *View* objektů a to, jelikož se jedná o mobilní zařízení s omezenými výpočetními prvky, není žádoucí.

Pro složitější kompozice prvků bylo použito dalších z předpřipravených layoutů, a to *RelativeLayout*. I zde název napovídá o hlavní vlastnosti tohoto layoutu, což znamená, že zde se vnitřní *View* objekty umísťují relativně vzhledem k layoutu samému. I zde je možnost vkládat jako jednotlivé prvky jiné *RelativeLayout* objekty. Použití tohoto layoutu již není tak jednoduché jak u LinearLayout, ale umožňuje vytvářet velmi efektivně i složitější kompozice prvků.

Jednotlivé prvky mají pomocí atributů nastavené vlastnosti. Příklad prvku umístěného v *RelativeLayout*:

```
<TextView
   android:id="@+id/lbl_closest_departure"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBaseline="@id/lbl_place_of_arrival"
   android:layout_alignParentRight="true"
   android:layout_marginRight="20px"
   android:text="@string/lbl_closest_departure"
   android:textStyle="bold"
   android:textSize="10px"
   android:visibility="invisible" />
```

U každého prvku včetně reprezentací *ViewGroup* je nutné uvést dva atributy - atribut `android:layout_width` a `android:layout_height`. Tyto atributy nastavují jak daný element bude široký a vysoký. U těchto elementů jsem používal předdefinované konstanty

[2] Jednoduché nástroje ulehčující práci

`wrap_content` určující minimální velikost pro plné zobrazení nebo `fill_parent`, která vyplní rodičovský prvek. Pomocí atributu `android:id` se nastaví identifikační kód daného prvku, na který se pomocí generované třídy *R* dá odkazovat.

Kapitola 5

Testování aplikace

5.1 Emulátor

Po dobu vývoje jsem aplikaci testoval na dobře propracovaném emulátoru systému Android, který je součástí SDK (viz 5.1). Tento emulátor umožňuje simulovat téměř vše, co lze na reálném zařízení. Měl jsem tedy možnost i vyzkoušet odesílání SMS z jednoho emulátoru na druhý. Hlavní nedostatek je v tom, že se jedná o prostředí emulované, tudíž není tak výkonné jako reálné prostředí na fyzickém zařízení.

Obrázek 5.1: Emulátor přibalený v SDK pro Android

5.2 Reálné zařízení

V průběhu vývoje byly některé verze důkladněji testovány na vlastním reálném zařízení. Hlavní výhody tohoto testování spočívaly v odhalování výkonnostních nedostatků aplikace. Aplikace byla následně také testována na dvou jiných zařízeních s různými vyššími verzemi systému Android od společnosti HTC. Konkrétně šlo o verze 1.5, 1.6 a 2.1 2.1.

5.3 Distribuce přes Android Market

Pro přístup do Android Marketu společnosti Google je nutné složit vstupní poplatek $25. Po zaplacení je zpřístupněno rozhraní pro nahrávání vlastních aplikací, které jsou pak distribuovány mezi uživatele pomocí aplikace Android Market v systému. Vývojář má možnost vložit jak aplikaci tak náhledy jednotlivých obrazovek, které se pak uživatelům v novějších verzích aplikace Android Market přímo zobrazují. Dále má vývojář možnost omezit v jaké zemi bude aplikace distribuována či nastavit popisky v různých jazykových mutacích. Systém dává vývojáři jednoduchou zpětnou vazbu o užívání dané aplikace. Je zde zobrazen celkový počet instalací a počet aktivních instalací. Aktivní instalace znamená, že uživatel aplikaci nainstaloval, ale doposud neodinstaloval a má jí stále na svém zařízení. Dále je zde k dispozici přehled hodnocení aplikace. Jednotlivé aplikace je možné stáhnout z distribuce či smazat úplně.

Pokud uživatel má nainstalovanou starší verzi a vývojář nahraje novou verzi na Android Market, při prohlížení aplikací se zobrazí uživateli informace, že existuje nová verze pro danou aplikaci a nabídne možnost si ji nainstalovat. V dalších verzích aplikace Android Market se nejspíš již objeví možnost automatického inovování aplikace, pokud se objeví nová verze [4].

Kapitola 6

Závěr

Cílem této práce bylo vytvořit aplikaci zjednodušující rezervaci autobusových jízdenek společnosti Student Agency pomocí mobilního telefonu. Bez aplikace je nutné zadávat poměrně složité formáty SMS zpráv, což jsem se pokusil v této práci eliminovat.

Přínos práce

Tato práce splnila mé osobní očekávání, neboť se mi podařilo vytvořit použitelnou mobilní aplikaci, která zjednodušuje často využívané operace okolo autobusových spojů společnosti Student Agency. V průběhu této práce jsem nahlédl do vývoje mobilních aplikací pro jednu z moderních a rychle se vyvíjejících platforem a současně okusil programovací jazyk Java. Osvojil jsem si využívání prostředků pro vývoj aplikací pro Android, které jsem již v průběhu psaní použil u jiných osobních projektů.

Během práce jsem narážel na různé problémy, které se v řadě případů podařilo vyřešit, jiné bohužel ne. Hlavním problémem byl nečekaně odmítavý přístup společnosti Student Agency ke spolupráci. Neměl jsem tedy žádný přístup k datům týkajících se jízdních řádů a správy rezervací. Veřejně tyto data nejsou nikde pro programátory aplikací třetích stran k dispozici. Jelikož jsem z těchto důvodů musel jízdní řády přepisovat ručně, nejsou ve výsledné aplikaci obsaženy všechny. Doplnění ostatních spojů je prací jednoduchou leč časově náročnou. Je nutno podotknout, že toto je slabé místo práce, pokud ke změně dat ze strany dopravce. Jejich sledování je bez jeho spolupráce pro jejich vysoký počet takřka nemožné. Již v průběhu zpracování a testování reálných údajů se objevili nesrovnalosti v jízdních řádech.

Bohužel se v průběhu vývoje projektu na trhu objevila aplikace s obdobným zaměřením, což mělo lehce demotivující charakter. Ve srovnání s touto aplikací přináší má práce řadu vylepšení - jako například možnost prohlížení jízdních řádů, správu kreditové jízdenky a umožňuje i odesílání jiných SMS zpráv než jen rezervačních.

Budoucnost projektu

Možnosti tohoto nápadu nejsou zdaleka vyčerpané, aplikaci jako takovou je, myslím si, možné ještě hodně rozšířit. Velmi užitečné by jistě bylo přidání možnosti správy rezervací. Toto řešení by však zajisté vyžadovalo již oficiální podklady společnosti byť jen pro parsování odpovědních SMS zpráv.

Zatím má aplikace výhodu, že pracuje bez připojení k Internetu, možné by bylo udělat mobilní obdobu webového rezervačního systému pro Android. Jízdní řády by byly aktuální, vyhledávání volných spojů interaktivnější a uživatelé s mobilním připojením k Internetu by ušetřili za odeslané SMS zprávy.

Tyto a jiné rozšíření však dle mého názoru nemá cenu dělat, dokud nebude se společností vyjednána spolupráce. Na tato jednání bych se osobně zprvu zaměřil.

Literatura

[1] Google: Android Reference. *Android Developers*, 2010, [online]. [cit. 2010-01-25]. URL http://developer.android.com/reference

[2] Google: Android Resources. *Android Developers*, 2010, [online]. [cit. 2010-01-04]. URL http://developer.android.com/intl/fr/resources

[3] Google: Android SDK. *Android Developers*, 2010, [online]. [cit. 2010-02-02]. URL http://developer.android.com/intl/fr/sdk

[4] Google: Developer Console. *Android Market*, 2010, [online]. [cit. 2010-05-10]. URL http://market.android.com/publish

[5] Google: The Developer's Guide. *Android Developers*, 2010, [online]. [cit. 2010-01-04]. URL http://developer.android.com/intl/fr/guide

[6] Herout, P.: *Java - bohatství knihoven.* KOPP nakladatelství, druhé vydání, 2006, ISBN 9788072322091.

[7] Herout, P.: *Učebnice jazyka Java.* KOPP nakladatelství, druhé vydání, 2006, ISBN 9788072323180.

[8] Herout, P.: *Java: grafické uživatelské prostředí a čeština.* KOPP nakladatelství, druhé vydání, 2007, ISBN 9788072323289.

[9] Petřek, P.: Vývoj pro Android - I. *Zdroják*, May 2010, [online]. [cit. 2010-05-12]. URL http://zdrojak.root.cz/clanky/vyvoj-pro-android-i/

[10] STUDENT-AGENCY: Jízdenky. *STUDENT AGENCY*, rok neznámý, [online]. [cit. 2010-03-11]. URL http://www.studentagency.cz/jizdenky-1462/czech/

[11] STUDENT-AGENCY: SMS Rezervace. *STUDENT AGENCY*, rok neznámý, [online]. [cit. 2010-03-10]. URL http://www.studentagency.cz/sms-rezervace/czech/

[12] Switkin, D.: Simple access to barcode scanning in Android, via Intents rather than direct use of project code. *Google Code Project Wiki*, February 2010, [online]. [cit. 2010-05-01]. URL http://code.google.com/p/zxing/wiki/ScanningViaIntent

[13] Ursiny, M.: Student Agency SMS Rezervace pro Android. *UrsiMon's blog*, November 2009, [online]. [cit. 2010-02-13]. URL http://ursimon.musichall.cz/?p=528

[14] Zahradník, O.: SMS Jízdenka. *Inmite s.r.o. - firemní blog*, 2009, [online]. [cit. 2010-01-13].
URL http://blog.inmite.eu/labs/sms-jizdenka/